REMARKS

Present Status of the Application

The Office Action objected claims 8 and 13 because of informalities. The Office Action also rejected pending claims 1-2, 6-8 and 12-13. Specifically, the Office Action rejected claims 1-2, 6-8 and 12-13 under 35 U.S.C. 102(e), as being anticipated by Yong (U.S. Patent No.5,991,819) or Chin et al. (U.S. Patent No.6,247,102). The Office Action also indicated that claims 3-5 and 9-11 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form. Applicants have amended claims 8 and 13 to improve informalities and added a new claim 14 for further definition. After entry of the foregoing amendments, claims 1-14 remain pending in the present application, and reconsideration of those claims is respectfully requested.

Discussion of Allowable Subject Matter

The Office Action indicated that claims 3-5 and 9-11 would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Applicants appreciate this indication of allowable subject matter.

Discussion of Office Action Rejections As being anticipated by Yang

The Office Action rejected claims 1-2, 6-8 and 12-13 under 35 U.S.C. 102(e), as being anticipated by Yong (U.S. Patent No.5,991,819) ("Yong", hereinafter). Applicants respectfully traverse the rejections for at least the reasons set forth below.

Yang reference is related to "a symmetric **multiprocessor system** constructed from industry standard commodity components together with an **advanced dual-ported memory controller**." The multiprocessor system includes a processor bus; **up to four processors** connected to the processor bus; an I/O bus; a system memory; and **a dual-ported memory controller** connected to the system memory. The dual ported memory controller has a first port connected to the processor bus to manage processor to system memory transactions and a second port connected to the I/O bus to manage I/O transactions. Two such systems can be connected together through a common I/O bus, thereby creating an eight-processor SMP system. **Yang reference is totally different from the present invention**, which can be easily found with reference to Fig. 2 of the Yang.

As shown in Fig.2, the system of the Yang includes up to four processors 201 connected to a high-bandwidth split-transaction processor bus 203. A system memory 205 is connected to bus 203 through an advanced dual-ported memory controller 207. The processor bus 203 is connected to the first port of memory controller 207. The second memory controller port connects to a I/O bus 215, also referred to herein as an expansion bus, which provides connection for multiple PCI I/O interfaces 209.

The advanced memory controller (AMC) 207 manages control and data flow in all directions between the processor bus 203 and I/O bus 215. The AMC 207 also controls access to a coherent DRAM memory array. The four processors use a bus snooping protocol on the processor bus 203. Bus snooping is a method of keeping track of data movements between processors and memory. If a processor needs data that is available in the data cache of another processor on the same bus, the data can be shared by both processors. Otherwise, the data must be retrieved from main memory 205, a more time consuming operation which requires system bus traffic. This method enhances system performance by reducing system bus contention.

However, the present invention is to provide a control chip having a multiple-layer defer queue and a method of operating the control chip so that a plurality of first type requests issued from a central processing unit (CPU) can be processed concurrently. Furthermore, **<u>the multiple-layer defer queue can support retry and defer transaction at the same time</u>** so that the amount of data flow between the CPU and the control chip is greatly reduced and overall performance of the system is improved.

In addition, the invention also provides a control chip with multi-layer defer queue, coupled to a CPU bus and a PCI bus. The control chip includes a PCI request queue, which receives a CPU request from the CPU bus, and generating a PCI request record. **<u>A multi-layer defer queue respectively responds to the CPU bus by one of a defer response and a retry response when the CPU request is received.</u>** A PCI access queue is used to receive the PCI request record. Also and, a PCI controller receives the request from the multi-layer defer queue and causes the PCI request record of the PCI access queue to be transmitted to the PCI bus via the PCI controller. Wherein, **when the PCI bus generates a response data and if the CPU request in the multi-layer defer queue is to produce the defer response, then the response data is directly sent to CPU bus. If the CPU request in the multi-layer defer queue is to produce the retry response and the CPU bus issues the CPU request, then the response data is directly transmitted to the CPU bus**.

For a proper rejection of a claim under 35 U.S.C. Section 102(e), the cited reference must disclose all elements or steps of the claim. Independent claims 1, 7 and 13 are allowable for at least the reasons that Yong does not disclose, teach, or suggest all of the features in claims 1, 7 and 13 above. More specifically, Yong at least does not disclose, teach, or suggest "*issuing a defer response or a retry response with respect to the request to the first bus*" and "*providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request*" as claimed in claim 1, or Yong at least does not disclose, teach, or suggest "*storing a plurality of requests issued from the first bus in the multiple-layer defer queue, wherein each of the requests has corresponding one response selected from the group consisting of a retry response and a defer response to be responded to the first bus*" and "*providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request*" as claimed in claim 7, or Yong at least does not disclose, teach, or suggest "*a PCI request queue for receiving a CPU request from the CPU bus, and generating a PCI request record*", "*a multi-layer defer queue, when receiving the CPU request, respectively* *responding to the CPU bus by one of a defer response and a retry response"* and *"when the PCI bus generates a response data and if the CPU request in the multi-layer defer queue is to produce the defer response, then the response data is directly sent to CPU bus, if the CPU request in the multi-layer defer queue is to produce the retry response and the CPU bus issues the CPU request, then the response data is transmitted to the CPU bus*" as claimed in claim 13.

For example, as asserted in the Office Action, the description "loading the request into a Pending Buffer to be deferred or retried" defined in the Col.9, Lines 42-52; Col.16, Lines 37-67; and Col.17, Lines 1-31 anticipated the "issuing a defer response or a retry response with respect to the request to the first bus." However, in the Yong reference, as described in Col.9, Lines 42-52, every transaction on the CPU bus is initially loaded into the PORQ, and the PORQ is actually implemented as **two parallel queues**, one targeted for local decode requests (Local PORQ), and one targeted for remote decode requests (Remote PORQ). **Both queues are loaded in parallel**, but only one (or none) of the two is "pushed" (marked as valid). ....**Requests "popped" off of the Local PORQ are routed either to the DRAM controller and/or the Pending Buffer Logic block**".

The Local Pending Buffer, as defined in Col.16, Lines 37-67, is as followed:

> The Local Pending Buffer maintains the coherency of transactions that target local memory that require a MIC subaction. This includes CPU Bus transactions for which a remote MIC is issued on the I/O Bus; and I/O Bus transactions for which a local MIC is issued on the CPU Bus (note that the MIC address always decodes to memory that is local to the AMC that generates it, however, local and remote in this context refer to the CPU Bus to which the MIC is targeted for issue). This block allocates a buffer entry when a MIC is generated and tracks the MIC completion. MIC completion occurs when the MIC is seen at the head of the ORQ of the target bus and snoop results are returned for the MIC. **Upon MIC completion, either a Deferred Reply is generated to he issued on the CPU Bus (thus completing the**

> **CPU Bus transaction that was deferred because a remote MIC was required), or an In-Order Response is given on the I/O Bus (thus completing the I/O Bus transaction that initiated the local MIC)**. 'Me Pending Buffer maintains coherency of these transactions by attempting to retry subsequent colliding transactions issued on either the CPU or I/O Bus. The Pending Buffer will not always be able to successfully force a colliding transaction to be retried. The exceptions occur when a HITM is asserted during the snoop phase which forces in-order completion of that transaction. These exceptions are described in detail in Section I 1, Handling Transaction Collisions.

The Remote Pending Buffer, as defined in Col.16, Lines 37-67 and Col.17, Lines 1-31, is as followed:

> The Remote Pending Buffer maintains the coherency of transactions that target remote memory that are issued and deferred on the CPU Bus. This block allocates a buffer entry when a remote targeted transaction is issued by a CPU and deferred on the CPU Bus. The block then generates a corresponding transaction to be issued on the I/O Bus and tracks the completion of this transaction. **Upon completion on the I/O Bus, the block generates a Deferred Reply to be issued on the CPU Bus**, thus completing the CPU Bus transaction that was initially deferred. Remote writebacks (either implicit or explicit), although not deferrable, will cause an allocation in the Remote Pending Buffer. The transaction will complete in order on the CPU Bus, but it is still the responsibility of this block to maintain the coherency of these writebacks until their completion on the I/O Bus. Essentially, these writebacks are posted by the Remote Pending Buffer. Coherency is maintained in a similar manner to that described above for the Local Pending Buffer. Again refer to Section I 1, Handling Transaction Collisions for more details.

As clearly defined above, in the Pending buffer, "**upon completion on the I/O Bus, the block generates a Deferred Reply to be issued on the CPU Bus**" which is different from "*issuing a defer response or a retry response with respect to the request to the first bus*", as defined in the claims of the invention.

In light of the aforesaid discussion, reconsideration and withdrawal of the Examiner's rejection is respectfully requested.

Discussion of Office Action Rejections As being anticipated by Chin

The Office Action rejected claims 1-2, 6-8 and 12-13 under 35 U.S.C. 102(e), as being anticipated by Chin et al. (U.S. Patent No.6,247,102) ("Chin", hereinafter). Applicants respectfully traverse the rejections for at least the reasons set forth below.

Chin reference is related to "a computer system employing memory controller and bridge interface permitting concurrent operation." The computer system includes a CPU, a memory device, two expansion buses, and a bridge logic unit coupling together the CPU, the memory device and the expansion buses. The bridge logic unit generally routes **bus cycle requests from one of the four buses to another of the buses while concurrently routing bus cycle requests to another pair of buses**. **The bridge logic unit preferably includes four interfaces**, one each to the CPU, memory device and the two expansion buses. Each pair of interfaces are coupled by at least one queue; write requests are stored (or "posted") in write queues and read data are stored in read queues. Because each interface can communicate concurrently with all other interfaces via the read and write queues, the possibility exists that a first interface cannot access a second interface because the second interface is busy processing read or write requests from a third interface, thus starving the first interface for access to the second interface. To remedy this starvation problem, **the bridge logic unit prevents the third interface from posting additional write requests to its write queue, thereby permitting the first interface access to the second**

interface. Further, **read cycles may be retried from one interface to allow another interface to complete its bus transactions**.

Chin reference is totally different from the present invention, which can be easily found with reference to Fig. 2 of the Chin. In the Chin reference, the bridge logic unit prevents the third interface from posting additional write requests to its write queue, thereby permitting the first interface access to the second interface and read cycles may be retried from one interface to allow another interface to complete its bus transactions.

However, the present invention is to provide a control chip having a multiple-layer defer queue and a method of operating the control chip so that a plurality of first type requests issued from a central processing unit (CPU) can be processed concurrently. Furthermore, **the multiple-layer defer queue can support retry and defer transaction at the same time** so that the amount of data flow between the CPU and the control chip is greatly reduced and overall performance of the system is improved. Furthermore, invention also provides a control chip with multi-layer defer queue, coupled to a CPU bus and a PCI bus. The control chip includes a PCI request queue, which receives a CPU request from the CPU bus, and generating a PCI request record. **A multi-layer defer queue respectively responds to the CPU bus by one of a defer response and a retry response when the CPU request is received.** A PCI access queue is used to receive the PCI request record. Also and, a PCI controller receives the request from the multi-layer defer queue and causes the PCI request record of the PCI access queue to be transmitted to the PCI bus via the PCI controller. Wherein, **when the PCI bus generates a response data and if the CPU request in the multi-layer defer queue is to produce the defer response, then the**

response data is directly sent to CPU bus. If the CPU request in the multi-layer defer queue is to produce the retry response and the CPU bus issues the CPU request, then the response data is directly transmitted to the CPU bus.

For a proper rejection of a claim under 35 U.S.C. Section 102(e), the cited reference must disclose all elements or steps of the claim. Independent claims 1, 7 and 13 are allowable for at least the reasons that Chin does not disclose, teach, or suggest all of the features in claims 1, 7 and 13 above. More specifically, Chin at least does not disclose, teach, or suggest "*storing a request in the **multiple-layer defer queue**, wherein the request is issued by the first bus*" and "*providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request*" as claimed in claim 1, or Chin at least does not disclose, teach, or suggest "*storing a plurality of requests issued from the first bus in the multiple-layer defer queue, wherein each of the requests has corresponding one response selected from the group consisting of a retry response and a defer response to be responded to the first bus*" and "*providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request*" as claimed in claim 7, or Chin at least does not disclose, teach, or suggest "*a PCI request queue for receiving a CPU request from the CPU bus, and generating a PCI request record*", "*a multi-layer defer queue, when receiving the CPU request, respectively responding to the CPU bus by one of a defer response and a retry response*" and "*when the PCI bus generates a response data and if the CPU request in the* *multi-layer defer queue is to produce the defer response, then the response data is directly sent to CPU bus, if the CPU request in the multi-layer defer queue is to produce the retry response and the CPU bus issues the CPU request, then the response data is transmitted to the CPU bus*" as claimed in claim 13.

In the Office Action, it is asserted that "a deferred queue storing CPU-to-PCI transaction cycles (Elements 142, 144 and 146 of Figure 3, Col. 30, Lines 57-67 and Col.31, Lines 1-12)" anticipated the "storing a request in the **multiple-layer defer queue**" or "storing a plurality of requests issued from the first bus in the **multiple-layer defer queue**, wherein each of the requests has corresponding one response selected from the group consisting of a retry response and a defer response to be responded to the first bus." However, the Elements 142, 144 and 146 of Figure 3 are respectively "CUP BUS MASTER STATE MACHINE", "DEFERRED QUEUE" and "SNOOP CONTROL", which are different from the **multiple-layer defer queue** of the invention.

Furthermore, in the Office Action, it is asserted that "responding to the CPU-to-PCI transaction cycles with either defer response or non-defer (retry) response (Col. 30, Lines 57-67 and Col.31, Lines 1-12)" anticipated "providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request" or "providing the responded data to the first bus if the defer response issues to the first bus; and providing the responded data to the first bus if the retry response issues to the first bus and only when the first bus again issues the request."

However, in the Col. 30, Lines 57-67 and Col.31, Lines 1-12, it is stated as follows:

> "In another aspect of the invention, the bridge device 104 may need to abort a transaction cycle. For example, in the following cases there is a non-deferred cycle in the P2IQ 184 (FIG. 2) or I2PQ 186:
>
> (1) A snoop request from the PCI interface 160 or AGP interface 150 is received by the processor interface. A CPU cycle is retried to avoid a deadlock;
>
> (2) The cycle after the non-deferred read is a read or a write from main memory 106 and the data is ready to be sent to the requesting device. The CPU 102 is retried to avoid slowing down the read or write cycle; and
>
> (3) A deferred CPU read has completed and the CPU interface 130 is ready to ship the data back to the CPU 102. Again, the CPU 102 is retried to get the data back to the CPU."

It is not anticipated that "providing the responded data to the first bus **<u>if the defer response issues to the first bus</u>**" as claimed in the invention.

In light of the aforesaid discussion, reconsideration and withdrawal of the Examiner's rejection is respectfully requested.

Application No.: 10/065,378 **Docket No.: JCLA6435**

CONCLUSION

For at least the foregoing reasons, it is believed that the pending claims 1-13 are in proper condition for allowance. If the Examiner believes that a telephone conference would expedite the examination of the above-identified patent application, the Examiner is invited to call the undersigned.

Respectfully submitted,
J.C. PATENTS

[illegible]

Jiawei Huang
Registration No. 43,330

Date: 8/23/2004

4 Venture, Suite 250
Irvine, CA 92618
Tel.: (949) 660-0761
Fax: (949) 660-0809